# 位相情報空間型データベースシステム (データ・セマンティクスの理論と実際に関する研究)

AUTHOR(S):

打浪, 清一; 手塚, 慶一

CITATION:

打浪, 清一 ...[et al]. 位相情報空間型データベースシステム (データ・セマンティクスの理論と実際に関する研究). 数理解析研究所講究録 1982, 461: 288-321

ISSUE DATE:

1982-06

URL:

http://hdl.handle.net/2433/103126

数理解析研究所講究録
461巻 1982年 288-321



# 位相情報空間型データベースシステム

阪大 工　打浪清一
手塚慶一

## 〔I〕はじめに

意味の包含，遠近関係を記述する言語モデルとして，位相情報空間モデルを提案し，そのFormalなモデル化とその数学的性質について検討してきた。またこれをデータモデルに拡張し，意味的な遠近，包含関係を記述，処理できるDBMSの構成について検討してきた。学術研究用のDBMSは，既知知識の管理用と，生データの整理・管理用の2種類が必要である。筆者らは提案したデータモデルに基ずくPrivate Research Data Management Systemとして Conceptual Schema-free Data Base Management Systemを提案，その試作を行ってきた。もう一つの Center Data Management Systemについては次のように考える。すなわち位相情報空間モデルにおける位相情報空間は知識のユニバースを表し，辞典・辞書として働くとともに，各Private Research File間の

インターフェースとる為の共通軸を提供する。

本稿では、これら二種のデータベースを統合するシステムの構成とその言語についてのべる。

位相情報空間は、意味地図であって、包含される概念は包含する概念の部分空間として表現され、意味的に近い概念は、空間中の近い部分空間として表現される。包含関係はある意味では集合関係ともみられる。一般に遠近と包含の二つの性質を同時に記述、処理することは難しく、この二者を十分取り扱えるDBMSはまだ見られない。本モデルはこの二つの性質の記述、処理を同時に行うことを目的としている。

位相情報空間モデルにおいては、ある視点を定めると、そのレベルでの認識対象としての個体が定まる。その個体はそのレベルでの位相情報空間内の点あるいは領域を占める。その空間を構成する各座標軸はその個体の属性を表すが、それが一段下のレベルの個体であることも多い。そして個体の点を各座標軸に射影したものが各属性の属性値となる。個体の認識レベルは相対的なものであり、P. Chenの個体・関係性モデルにおける関係性に対応するものも、視点をそこに移せば個体と考えられる。視点を置かれたレベルに対し、これら個体間の関係として生じた事象は関係となる。よって関係ー個体ー属性ー属性値、という階層ないし関係は相対的なもの

で，現在視点がおかれているところが，個体レベルとなる。

## [II] 位相情報空間モデル概説

生成意味論の立場でモデル化している。位相情報空間生成文法を基本規則として定義する。この文法により生成された空間を位相情報空間とよび，この空間が意味地図となる。位相情報空間生成文法から統辞生成文法を求める写像が存在し，この写像を用いることにより各概念領域はそれらが表す語，句，または文に変換される。

表層の統辞文法では，生成された文の間に何等直接的な意味関係は陽に表れていないので，ただ単に文の集合を生成すればよいが，空間生成文法では点あるいは領域の生成だけではなく，その位相も併せて生成しなければならない。ところが使用時には，位相ではなく特定概念領域を指示すればよいから，結局生成規則には空間全体の知識構造を生成するものと，伝達したい個々の概念を生成するものと2種類ある。ユニバース生成規則は位相空間全体（構成要素集合とその上での位相）を生成し，発話・意味解釈時には概念生成規則で特定概念領域を生成する。

位相情報空間の各座標軸は何れもが同じ位相としているとは限らない。そこで空間生成文法で生成する位相に関して，

教育漢字約1000字の表現する意味について分析した結果、次のような位相の種類が少くとも必要であることが分った。

(1)数軸　例えば身長、体重のように実数値あるいは整数値などで表現されるもので、座標軸は、R、N、あるいは適当な両閉区間、開区間等で定義される。

(2)循環数軸　例えば月の名前：1月、2月、…、12月、1月、…のように数値で表現され、その値が循環するもの。これは(1)の数軸の両端点を同一視することにより定義される。

(3)順序軸　例えば、温度感覚軸：寒い、涼しい、快適、暖い、暑い、…のようにその性質が順序づけられた序列として表わされるもの。この軸では順序のみが意味を持ち、距離は考えない。これは要素の順序づけした列挙法により定義される。

(4)循環順序軸　例えば　四季：春夏秋冬春…のようにその性質が循環する序列で表わされるもの。これは(3)の両端点を同一視することにより定義される。

(5)より一般的な位相　例えば　赤、青、黄、緑、…のような色の関係のように何等かの位相的な関係があるが、上記(1)～(4)では表現できないもの。これらは何等かの位相幾何学的方法により定義される。

(6)単なる集合　位相関係が無く，単なる寄せ集めのもの。

よって位相情報空間生成文法においては，空間生成時にそれぞれの属性の上記タイプに合わせて，それに見合った位相の座標軸を生成してやくように文法を定める。

位相情報空間生成文法は，生成規則の適用に制限をつけた次の3レベルの文法からなる。

[定義1]　空間生成文法　$G_1$

$$G_1 = < V_{N1}, V_{T1}, P_1, S_1 > \quad (1)$$

ここで　$V_1 = V_{N1} \cup V_{T1}$,　$V_{N1} \cap V_{T1} = \phi$　(2)

$V_{Ni}$, $V_{Ti}$, $P_i$, $S_i$ は第 $i$ レベルの中間語彙，終端語彙，生成規則，初期語彙を表す。また他レベルでも(2)に対応する式が成立するとする。

$P_1$ は次の形の規則からなる。

$$P_1 : \quad A \to w \qquad A \in V_{N1},\ w \in L^{\Delta}(V_1, D) \quad (3)$$

$$L^{\Delta}(V, D) \stackrel{def}{=} \{ (\alpha \delta)^* \beta \mid \alpha, \beta \in V,\ \delta \in D \mid \quad (4)$$

$$D = \{ \cdot, \#, | \} \quad (5)$$

$G_1$ は情報空間を構成する部分空間間の関係を規定するもので，Dは空間構成子で，その種類としては，直積・，連結和#，直和|　がある。

[定義2] 空間性質規定文法 $G_2$

$$G_2 = < V_{N2}, V_{T2}, P_2, S_2 > \quad (6)$$

$P_2$は次の形の規則からなる。

$$P_2 : A \to w \qquad A \in V_{N2} \qquad a, b \in V_{T2} \quad (7)$$

$$w \in \{ a, \dot{a}, aa, aa^{-1}, aba^{-1}b^{-1}, aba^{-1} \} \quad (8)$$

$G_2$は各部分空間の性質を規定するもので，その性質はwの要素で定まるが，その種類には，

(i) 直線軸 (a) $\mathbb{R}, \mathbb{N}, \mathbb{Z}$ またはその区間

(ii) 円 ($\dot{a}$) (i)の直線軸の両端点を同一視したもの．

以下位相幾何学の多様体を表す多角形式の

(iii) 射影平面 ($aa$)

(iv) 球面 ($aa^{-1}$)

(v) 一冗の輪環面 ($aba^{-1}b^{-1}$) トーラス

(vi) 一冗面 ($aba^{-1}$)

等の制限型がある。このレベルで座標軸を組み合わせて部分空間を構成する方法を定める。

[定義3] 位相規定文法 $G_3$

$$G_3 = < V_{N3}, V_{T3}, P_3, S_3 > \quad (9)$$

$P_3$は知識全宇宙を生成する$P_{3T}$と，空間内の領域を生成す

3 $P_{3A}$ とからなる。

$$P_{3T}: \quad A \to w \qquad w \in \{\text{各型の位相空間}\} \tag{10}$$

$$P_{3A}: \quad A \to w \qquad w \in \{\text{位相空間の領域}\} \tag{11}$$

$G_3$は部分空間の位相を与えるもので，その位相の型としては，次のようなものがある。

$T_0$, $T_1$, $T_2$, R, $T_3$, CR, T, N, $T_4$, CN, $T_5$, FN, PN, $T_V$, L, C, CH, LC, $C_n$, U, S, QM, M, B, ...

以上の3レベルを統合して，位相情報空間が生成される。

[定義4] 位相情報空間生成文法 $G_I$

$$G_I = < G_1, \ G_2, \ G_3 > \tag{12}$$

ここで 3レベルの間の関係は次のように定められる。

$$S_1 = S, \qquad S_2 = V_{T1}, \qquad S_3 = V_{T2} \tag{13}$$

各レベルの適用規則に制限をつけることにより，種々の空間が構成される。

[定義5] 位相情報空間 I

$$I = I(G_I) \tag{14}$$

位相情報空間生成文法 $G_I$から生成された位相空間Iを，位相情報空間という。このとき第3レベルでは，$P_{3T}$のルー

ルを用いる。この空間が言語モデルにおいては基礎をなす。

データモデルのデータベースへの応用においては，この空間が Conceptual Schema に対応し，位相情報空間生成文法が，この Schema を記述するのに用いられる。

## 〔III〕位相情報空間型データベースシステムでのデータ表現

データベースで取り扱いたいデータの種類には，宣言的知識，手続的知識がある。本システムではこの二者を次のように扱う。

宣言的知識：概念間の遠近，包含関係をもとに，その関係を反映する形で空間内の領域として配置する。

手続的知識：手続間の遠近，包含関係（定義域，値域の包含関係，計算機能の包含関係）をもとに，その関係を反映する形で空間内の領域として配置する。手続そのものは，その領域にストアされる。

取り扱いたいデータは，別の観点から，内包的知識と外延的知識に分けられる。この取り扱いは次のように考える。

内包的知識：情報空間内に配置されている各概念（点あるいは領域）に対し，各座標軸へ射影し，その

座標値を求めることが，内包を知ることに相当する。

外延的知識：集合あるいは領域が与えられたとき，それに含まれる要素概念（点集合）を求めること，また逆向きの処理が相当する。

本システムはこのように 宣言的知識，手続的知識，および，内包的知識，外延的知識をカバーするように考えられているが，どちらかというと宣言的知識を主に考えている。これに対し手続的知識を主として考えると，S-M変換可能な言語をベースとするシステムの方が処理が行いやすい。その様なシステムについては，次の「外延と内包を取り扱える拡張データベースシステム」で議論する。但しまだ予備的なので手続きの遠近を反映する形での格納はまだインプリメントされていない。

データベースに貯えるデータ群の特性について，2つの構造が考えられる。Inter-entity structureと，Intra-entity structureである。

この2つのデータ構造の取り扱いは次のように行なう。

Inter-entity structure：個体間構造

データの一つ一つが構造が異るとき，一つのデータを記録するのに一つスキーマが必要となり，効率が悪い。このとき構造を生成する文法を定義し，この文法を用いて構造を生成し，その生成樹（Generation Tree又はDerivation Tree）を抽録とする。これを使用生成規則，使用終端語彙で表現し，データベース化したものが，「内容検索可能な画像データベースシステム」であり，そのパイロットシステムとして，植物図鑑データベースシステムを構成し，その有効性を確認した。

この段階では，内部構造の抽録であるが，各個体の内部構造の類似性を調べて，個体間の関係，類似性をもめれば，Inter-entity structureは，次のIntra-entity structureに変換できる。

Intra-entity structure：個体内構造

データ群は同じ構造をしており，内容（属性）のみが異るとき，構造をスキーマで記述し，個々のデータはOccurrenceとして取り扱う。これは従来のやり方のDBMSで実現可能である。

位相情報空間型データベースシステムでは，個体から始まり，集合，線，領域，更にこれらが再帰的に入ったものとして，データタイプが定義される。これらは個体内構造であり各要素間の包含，近接関係などは個体間構造である。

［定義6］ 個体

実体を持つか，あるいは概念上のもので，他のものと区別して示されうるものをいう。位相情報空間内の一点もしくはある領域を占める。それ故各個体には幾つかの属性（空間の座標軸に対応）が付属し，更にその属性の属性値が射影により定まる。

［定義7］ 集合

ある個体（更にはRecursiveに集合等も入り得るか）の集合がAbstractionにより一段上のレベルの一つの個体とみなされたとき，これを集合という。集合においても，属性，属性値が付随するか，属性値が個体等となる場合もある。

［定義8］ 線

ある個体（更にはRecursiveに集合等も入り得るか）の順序づけられた組が抽象化されて一つの個体とみなされたとき，これを線という。最も下位レベルの線は，各座標軸上に並んだ属性値の線である。

［定義9］ 領域

位相情報空間内の（閉）部分空間を領域という。これは一般に個体の Generalization によってできた個体に相当する。一般に概念の階層構造は，個体と領域の関係として表現される。即ち領域が類概念，個体が種概念に対応する。この関係も Recursive に定義されるので，視点が変われば種概念が類概念となり更に下位レベルの種概念が定義され，又逆に上位方向への視点の移動もあり得る。

ある種概念の内包は，直接上位の類概念の内包に種差を加えたものに等しい。ここで種差とは，ある種概念と同一の類概念に属する他のすべての種概念から区別する特徴，即ち当の種概念の特有の性質をいう。それ故領域と，その領域を構成する個体集合に対し，領域の内包即ち各属性の属性値の組に，種差の属性を加えると，個体の各属性の属性値の組が得られる。換言すればある軸を無視して出来る部分空間は，Generalization による領域を構成している。

これに対し Aggregation は，属性から空間を構成し個体を形造るプロセスに相当する。換言すれば各属性は個体の部分を表している。

Role は個体の一属性として入ってくるのが普通であるが、

それが抽象化されて個体として取り扱われることもある。

M.L. Brodie の Association is member の relationであるが…これは個体集合とRoleで一つの個体を作ることに相当する。例えば従業員の集合と、組合というRoleで職員組合という一段上の個体が作られる。

同一の類概念（領域）に含まれる複概念（部分領域あるいは個体）は互に同位概念となり、述言的概念、反対概念、矛盾概念等が考えられる。これらはGeneralizationを行って無視した属性軸において、その直和表現や、両端点の表現等に相当し、位相情報空間データベースではこれらの概念の導出も可能である。

一つの概念に別名がある場合、また換言した場合、これらは同一概念となる。このときその概念領域には複数個の表層の単語または句や節が対応する。例えば「両親」を表す概念領域は、「父母」の表す領域と等しい。

内包は異るが外延は等しい概念は等値概念というが、例えば、日本の現首相と鈴木氏というのがこれに当る。これは、前述のRoleの処理で記述できる場合が多いと考えられる。

## [Ⅳ] 位相情報空間型DBMSの言語

位相情報空間型DBMSには，学術研究用のPRDMSとしてのConceptual Schema-free DBMSと，CDBMSとしてのDBMSがある。これら2つの言語はできるだけ共通性があるのが望ましい。そこで共通的な骨格部分を現在設計中である。以下DDL，DMLについて概略をのべる。

本システムで取り扱うデータの対象としては，事務データのみならず，意味データの取り扱いに大きな重点をおいている。

### (1) 位相情報空間型DBMSのDDL

位相情報空間生成文法によって生成される知識ユニバース（$P_{3T}$ を用いて生成された空間）を記述する為には，最低限次のものを記述する機能が必要である。

空間と空間の合成法（空間と空間の間の関係）
空間の構成法（空間を構成する軸と，その組合せ方）
各軸の詳細

これらを記述する為のDDLとして次の2つのレベルを設ける。何れのレベルで記述してもよい。

(i) Conventional level

正確には全体のSyntax とBNFで記述すべきであるが、ここでは 大切なポイント部分のみを文脈が分りやすい形で示す。

## 各軸の詳細

```
PROPERTY <property name>,<property ID #> ; <comment>
      [ALIAS(<property alias>,)*]
       TOPOLOGY TYPE  <topology type>,
       TYPE  <data type>  <data expression length>
      [, RANGE  <range> ]
      [,<occurrence> ]
      [,KEY ][,NONULL ]
      [,DEFAULT  <default value> ]
      [,ORDERING  <sorting key> ]
      [,UNIT  <unit> ] :

<topology type>::= LINE | CYCLINE | ORDER | CYCLORDER |
                   GTOPOLOGY | SET
<data type>::= INTEGER | REAL | LOGICAL | CHARACTER | KANJI |
                   BIT
<data expression length>::= <integer>
<range>::= <lower limit> : <upper limit>
<lower limit>::= [<number> | (<number>
<upper limit>::=  <number>]|  <number> )
<number>::= <integer> | <real> | <TF-value> | <special value>
<default value>::= <number>
```

<sorting key>::= ASCENDING | DESCENDING | ALPHABETIC | AIUEO |
IROHA | <value list>

<occurrence>::= UNIQUE | MULTI | VARIABLE TIMES

<unit>::= mm | cm | m | km | g | kg | sec. | min. | hr. | year|
w | j. | cal | ...

<special value>::= Θ | Φ | Ω | *

## 空間の構成（個体の属性空間の構成）及び空間の合成

SPACE <space name> , <space ID #> ; <comment>
[ALIAS ( <space name>, )*]
SPACE TYPE <space data type>
[,<occurrence>]
[, UNIT <unit>]
(,<extended property name> [,<role>][,<key property>]
[,<sorting order>])
[ <space constitutor>
( <extended property name> [,<role>][,<key property>]
[,<sorting order>])]* :

<space data type>::= ENTITY | SET | LINE | AREA | ...

<key property>::= KEY | COMPOUND KEY <i>

<sorting order>::= [ ASCENDING / DESCENDING ] <i>

<extended property name>::= <property name> |
<entity name>[.<ID name>] <subspace name>[.<ID name>]

<space constitutor>::= # | | | ·

ここで <role> とは、空間中に表現される個体に対し、extended property のもつ役割をいう。

空間の構成は Recursive に行われる。低位のレベルでは空間を構成する要素として属性が入るが、次第に上位のレベルにゆくに従って、個体や部分空間を要素として空間が定義される。ある空間が2通り以上に分解されるとき、その空間は、各分解の積で定義される。例えば人間という空間は男、女という性別の部分空間に分割され、同時に年令別の部分空間に分割される。

これは例えば次のように記述される。

```
SPACE MAN, E1; NINGEN WO ARAWASU KUUKAN        空間 E1. 人間
    ALIAS  HITO, NINGEN,                       別名 →人
    SPACE TYPE ENTITY,                         空間型 → 個体
    UNIQUE,                                          → 一価
    UNIT NIN,                                  単位  →  人
    ( NAME, ID, KEY, AIUEO ) · ( SEX ) · ( AGE ) :   空間構成

PROPERTY NAME, P1 ; NAMAE                      属性  P1.  名前
    ALIAS NAMAE, SEIMEI,                       別名  →姓名
    TOPOLOGY TYPE ORDER,                       位相型 →順序型
    TYPE CHARACTER 20,                         型    →文字型 20字
```

```
    UNIQUE,                          → 一価
    KEY, NONULL,                     キー，ヌル値は許さず．
    ORDERING AIUEO :                 順序    → アイウエオ順

PROPERTY SEX, P2 ; SEIBETSU          属性  P2.  性
    ALIAS SEIBETSU, SEI,             別名    →性別
    TOPOLOGY TYPE ORDER,             位相型  →順序型
    TYPE CHARACTER 1,                型      →文字型1字
    UNIQUE,                                  →一価
    ORDERING [M,F] :                 順序    → M, F の順

PROPERTY AGE, P3 ; NENREI            属性  P3.  年令
    ALIAS NENREI, TOSHI,             別名    →年令．年
    TOPOLOGY TYPE LINE,              位相型  →線
    TYPE INTEGER 3,                  型      →整数3桁
    RANGE [0,150],                   値域    →[0，150]
    UNIQUE,                                  →一価
    ORDERING ASCENDING,              順序    →昇順
    UNIT SAI:                        単位    →才
```

Conceptual Schema の作成時には，これらの Syntax の前に，CREATE をつけて CREATE SPACE，CREATE PROPERTY のように書く。修正時には，これらの Syntax の前に MODIFY をつけて MODIFY SPACE，MODIFY PROPERTY のように書き，修正部分のみを記述する。

利用時における External Schema の作成時には，これらの Syntax の前に LET をつけ LET SPACE，LET PROPERTY のように書く。

(ii) Generative Rule Level

位相情報空間生成文法を陽に記述してゆく方法である。生成規則を記述し，その生成規則に表われる各語の定義を行う方法で記述する。記述内容は，Conventional Level とほぼ同じである。

```
<space name> → (<extended property name> [, role ])
              [ <space constitutor>
               (<extended property name> [, role ])]* :
WHERE <space name> IS <space ID #> ; <comment>
               [ ALIAS (<space name> , ) * ]
               [ SPACE TYPE <space data type>]
               [ , <occurrence> ]
               [ , UNIT <unit> ]:
    [ <extended property name> IS
               [ , <key property> ]
               [ , <sorting order> ] ]:

<extended property name>::=<subspace name>[.<ID name>] |
            <entity name>[.<ID name>]|<property name>
```

<space constitutor>::= # | | | ·

<space data type>::= ENTITY | SET | LINE | AREA | ...

<occurrence>::= UNIQUE | MULT | VARIABLE TIMES

<key property>::= KEY | COMPOUND KEY <i> | NON KEY

ここで アンダーラインは default値を表す。

<property name> → <topology type> <topology expression>

WHERE <property name> IS <property ID # >; <comment>

[ ALIAS ( <property alias> , ) * ]

[ , <occurrence> ]

[ , KEY ][, NONULL ]:

<topology type>::= LINE | CYCLINE | ORDER | CYCLORDER |

GTOPOLOGY | SET

<topology expression>::= <var.> | <var.i> <var.i>

<var.i> <var.j> <var.i>$^{-1}$<var.j>$^{-1}$|<var.i> <var.i>$^{-1}$|

<var.i> <var.j> <var.i>$^{-1}$

ここで <var.k> は 軸kを表す変数である。

<var.k> → <data type> <data expression>

WHERE <var.k> IS <axis ID # > ; <comment>

[ ALIAS ( <axis alias>, )*]

[ , <occurrence> ]

[ , KEY ][, NONULL ]:

```
<data type>::= <type> <length> <exp.unit>
<type>      ::=  INTEGER | REAL | LOGICAL | CHARACTER | KANJI |
                 BIT | ...
<exp.unit> ::=  BYTE | BIT | CHARACTER | ...
<data expression>::= RANGE <range>
              [ , DEFAULT <default value> ]
              [ , ORDERING <sorting order> ]
              [ , UNIT <unit> ]
<range>     ::= <lower limit> : <upper limit>
<lower limit>::= [<number> | (<number>
<upper limit>::=  <number>]|  <number> )
                 :
                 :
```

以上の → のついた3レベルの生成規則を定義して空間の構成を定め，その詳細は，生成された文および WHERE 句内で定義する。

## (2) 位相情報空間型DBMSのDML

位相情報空間生成文法により生成される概念スキーマを、DDLで記述したが、その上での処理を行う為のDMLには、他のモデルで持つ機能に加えて、次の機能が必要であり、またこの機能を持たすことにより、他のモデルではシステムがサポートすることのできない意味的な遠近、包含関係を取り扱う処理が可能となる。

①意味的に近い概念、遠い概念をアクセスする機能
②反対語、同義語、対語的な概念をアクセスする機能
③意味的に包含関係にある概念をアクセスする機能
④全知識が一つにまとまっているので、知りたいところだけを切り出すWindow的な機能
⑤Windowから眺めたデータを、どの視点から見るかというAspectを指定する機能

これらの機能は他のデータモデルに基づくデータベースでは、十分にサポートすることは、おそらく不可能である。

DMLも次のように 情報代数系レベル、Conventional level、自然言語風レベル、空間表示レベル等幾つかのレベルがある。

(i) ファイル情報代数系レベル

Conceptual Schema-free DBMSを設計した際に，データ統合の為に設定したファイル情報代数系を用いて記述するレベルである。

これは更に2つのレベルに分けられる。一つはファイル情報代数系を用いて欲しいデータを未知数とする方程式を記述し，システムがこれを解いて答えるレベルで，非手続的な高級なレベルである。他の一つはファイル情報代数系の性質を用いるが，処理はこの上で手続的にプログラムしたものを翻訳，実行するレベルである。このレベルの言語はここでは省略する。

(ii) Conventional level

積極的にExternal Schemaを記述し，これを用いて処理を行うレベルで，現在普通のDBMSが行っている方式によるものである。

このレベルでは，処理対象を個体とみなし，その属性，属性値の3つ組のうち，何れかを陽に与え，指定した部分を求める処理が主になる。この処理を行っているうちに視点がずれて関係性が入ってくる場合があるが，そのときは関係を個体とみなして，個体はその属性とみて処理を続けてゆく。

このレベルのデータ検索のコマンドの核部分を次に示す。

FIND <var. list> WHERE <conditional clause> ... query st.

<var. list>::= <var.> | <var.>, <var. list>

<conditional clause>::= <e.r.f> | <e.r.f> <l.op> <cond. exp.>

<e.r.f> ::= { e=<ext. data-item> / p=<ext. data-item> / v=<ext. data-item> } |

{ e=<ext.data-item> / p=<ext.data-item> / v=<ext.data-item> } ( <l.op> { <e= ext. data-item> / : / <p= ext. data-item> / : / <v= ext. data-item> } )+

*where e.r.f means entity restricting formulae*

<cond. exp.>::=<variable><r.op><variable>

<variable>::=<string>|<functional exp.>|<cond.exp>

<functional exp.>::=<function>(<expression>)

<function>::=<set function>|<entity function>|<line function>|
<area function>| ...

<set function>::= COUNT | MAX | MIN | ...

<line function>::= LENGTH | RANGE | DOMAIN | ...

<area function>::= AREA | DIAMETER | ...

<entity function>::= NO_OF_PROPERTIES | ...

<ext. data-item>::=[<determiner>]<data-item> [;<string>]

<l.op>::= AND | OR

<determiner>::= SOME | EACH | NOT | ALL

<r.op>::= < | ≦ | > | ≧ | = | ≠ | ⊃ | ⊂ | ≃ | ↔

Ex.

```
FIND d, AVG(x) WHERE { e=EACH dept;d  AND   y
                     { p=   emp       :   salary
                     { v=    y        :      x

FIND x WHERE { e=dept;z  AND       AND  item;y
             { p=item;y   :  floor  :    dept
             { v=  x      :    2    :      z
      AND COUNT(z) ≧ 2
```

・各部門の従業員の平均給料を求めよ。

・2階にある少くとも2つの部門によって売られている品目をすべて求めよ。

このレベルでは検索しては視点を移し、別の視点から検索し、更に視点を移す形で Traverse しながら検索を行ってゆく。これは $\langle e, r, f \rangle$ の中で変数名を共通にすることにより行う。$\langle e, r, f \rangle$ では常にその時点での処理対象を e, その属性, 属性値を p, v として表す。

r.op の ⊃, ⊂ は概念の包含関係を表し, ≒ は概念的な近似性を表し, ↔ は反対概念を表す。

関数には, その定義域により, 個体型, 集合型, 線型, 領域型のタイプがある。例えば LENGTH は線の長さを求める関数で, AREA は領域の面積を求める関数である。

(iii) 自然言語風レベル

External Schema は記述せず、質問文の中に適当に Schema 情報を入れて曖昧さをなくして処理を行うレベルである。

知識辞書として、辞書や辞典を位相情報空間生成文法に基いて解析し、データベースを構築し、それを利用する場合を想定する。この場合、毎回 External Schema を記述することは実際的でない。そこで自然語に近い形で質問をするか、曖昧さの生じそうな所では、External Schema に相当したデータを陽に記述しながら、質問を書くためのレベルである。DML で最もよく用いる検索に関する部分で、他のシステム、関係モデル型や、転置型DBMSでは記述しにくい、あるいは記述できない部分について重点的にのべる。

```
PRINT <var.list> FIND [<functional expression>                    ]
                      [<var.list> <restricting clause>            ]

<var.list>::= <var.> | <var.> , <var.list>
<var.>::= [ITS] [<determiner>] <e.e.p> [;<string>]{[<o.r> <var.>]}*
                                                  {[<l.op> <var.>]}
<determiner>::= ALL | SOME | EACH
<e.e.p>::=<extended entity> | <extended property>
<o.r>::= OF | RELATED TO
<l.op>::= AND | OR
```

<functional expression>::=<function exp.> (<arith.op> <var.>) |
<function exp.><arith.op> <function exp.>

<function exp.>::=<function>[<function exp.> | <query state.> | <var. list>]<range of object of function>

<function>::=<entity function> | <set function> |
<line function> | <area function>

<entity function>::= NO_OF_PROPERTIES | ...

<set function>::= COUNT | SUM | MAX | MIN | AVG | SORT | ...

<line function>::= LENGTH | DOMAIN | RANGE | AVG | ...

<area function>::= AREA | DIAMETER | ...

<range of object of fn.>::=<o.m.> <var.>|
BETWEEN (<value> ~ <value>) OF <ext. property>

<o.m>::= OF | AMONG

<restricting clause>::= WHERE <condition list> |
[WHOSE [<determiner>]<ext.property>]<restriction>
FROM STANDPOINT <view point> [AMONG | OF]<ext.entity>;<var.list>|
WHOSE [<determiner>]<ext.property><r.op><v.c.f.i.>
<l.op><conditional list> |
<substituting statement>

<restriction>::= CONTAIN | CONTAINED | CONSTITUTE | CONSTITUTED |
COVER | COVERED | BELONG | BELONGED | NEAR | CONTRARY |
COORDINATE | DISJUNCTIVE | CORRELATIVE | CROSS |
EQUIPOLLENT | IDENTICAL

<condition list>::=<condition> (<l.op><condition>)* |
NOT (<condition> ) <o.m> <var.>

<v.c.f.i>::=<var.> | <const.> | <function exp.> | <identifier>

<condition>::=<functional expression>|<var.><r.op><v.n.f.i>

<r.op>::= > | ≧ | < | ≦ | = | ≠ | ≃ | ≠ | ↔

<viewpoint>::= <ext.property>|<ext.property><viewpoint>

<substituting state.>::= ← <query statement>

<judge statement>::= IS IT TRUE <functional exp.> ? |

IS IT TRUE <var.list><restricting clause> ?

⋮

<restricting clause> が何重にも重なったときは、後から直前へかけながら括弧をつけてゆくように解釈する。

<function> には、位相情報空間モデルの各データタイプに対するものが存在する。他のDBMSでは <set function> しか無い。

<restricting clause> の右辺式の意味は、<ext. entity> を対象として、<viewpoint> で指定される観点から、換言すると、<ext. entity> が位置する空間中の <viewpoint> で指定される座標軸に関して <restriction> で与えられた条件で検索することを示している。

<restriction> は位相情報空間を読み命令であって、次のようなものがある。

CONTAIN(ED) は Generalization 関係を読み、<viewpoint> で指定される属性において、<ext. entity> を含む（に含まれる）ものを検索する。

BELONG(ED) は集合と要素の関係 ∈ (∋) にあるものの検索である。

CONSTITUTE(D) は Aggregation 関係を読み、<viewpoint> で指定される属性において <ext. entity> を aggregate した(された)ものを検索する。

COVER(ED) は、領域または集合における包含関係を検索する。例えば 技能A, B, C を cover する人をみつける。

NEAR は空間中 <ext. entity> に近いものを検索する。(<viewpoint> で指定される属性に関して)。

CONTRARY は空間中 <viewpoint> で指定される属性において対極にある反対概念を検索する。

COORDINATE は <viewpoint> で指定される属性レベルで <ext. entity> に属する同レベルの概念を全て検索する。

DISJUNCTIVE は COORDINATE とよく似ているが、互に素の概念の集合を検索する。

CORRELATIVE は <viewpoint> の観点で対となる相対立する概念を検索する。

CROSS は <viewpoint> 軸上で、<ext. entity> と交わる概念を検索する。

EQUIPOLLENT は <ext. entity> 空間中で <viewpoint> の観点から見て，換言すれば他の属性は無視して，内包は異るが，外延が同じものを検索する。

IDENTICAL は <ext. entity> 空間中で，<viewpoint> の観点から内包，外延とも同一のものを検索する。

これらの Restriction は次のように考える。

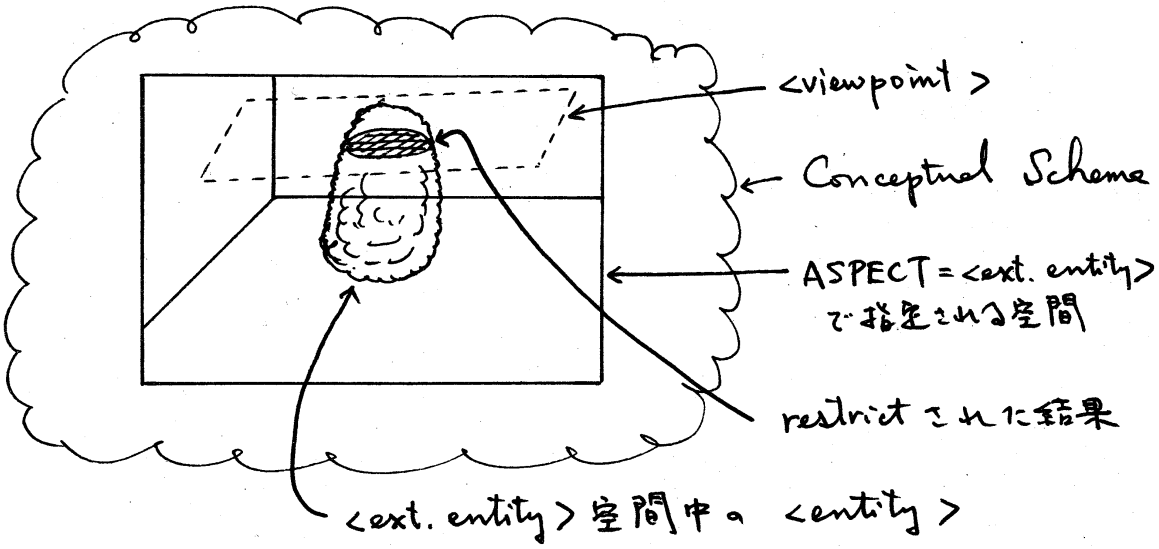


図1　Restriction 処理　（この Restriction は Relational Model の Restriction ではない！）

<range of object of fn.> のオー式は　OF <ext. entity> の形で使われ，<ext. entity> の属性を抽出する処理である。これに対し AMONG <ext. entity> は，個体集合中の個体を抽

出する処理である。即ちOFでは<ext. entity>から一つレベルの下ったものをさがすのにおし、AMONGでは<ext. entity>と同しベルのものをさがす点が異る。

<judge statement>は、真偽を問合せる命令である。

<ext. entity>は元表のentityであって、個体および一段上の関係が生じ、その関係に視点を移しそれを個体とみなしたもの　を指す。

<ext. property>は元表のpropertyであって、属性および一段上の関係が生じ、その関係に視点を移しそれを個体とみなしたときその個体の属性となるもの、換言すればもとの個体に対応するもの　を指す。

検索のFINDほかの　INSERT, DELETE, UPDATE　なども同様に定義されるが省略する。

Ex.

PRINT value FIND AVG( AVG( salary OF employee OF EACH dept.)
AMONG dept. )

各部門当りの平均給料の、全部門にわたる平均を求めよ。

PRINT name FIND employee WHOSE property COVER FROM STANDPOINT
skill AMONG employee　skill = A AND B AND C

A, B, Cの技能をカバーする従業員をみつけよ。

PRINT name FIND organs CONSTITUTE FROM STANDPOINT parts OF face

顔の構成器官名をあげよ。

PRINT name, address FIND student WHOSE address NEAR FROM
STANDPOINT distance AMONG students OF department OF
engineering, WHOSE address ≃ Umeda, Osaka

工学部の学生のうち、距離的にいって大阪市梅田に近い所に住んでいる者の 名前と住所を求めよ。

(IV)空間表示レベル

会話によるstring出力だけでなく、空間を指定した条件で1～3次元空間に射影し、それをブラウン管上に表示し、会話型で条件をしぼってゆくレベルである。位相情報空間はもともと高次元の地図なので、これを文の表現に直して出力したのでは情報の大切な部分が失われてしまうので、地図を地図のままで表示し、人がそれを読む方法である。

これは内容検索可能な画像データベースシステムとして作成したDBMSをmoditfyして動かす予定であるが本稿では省略する。

階層構造は、色分けや、Zooming, Panningなどにより階層を上下することにより表示する。

## [V] 位相情報空間型DBMS

本DBMSは前述の如く Private Research Data Base Management System と Knowledge Data Base Management System の統合システムとして構成される。

知識管理システムにおいては、データは生成規則に基ずき多次元の Facet に分割される。ファイル構成は、転置ファイル索引をもつ Clustered Index Multi-key Bucket Resolved File 構成をとる。Aggregation 関係は、key 索引とデータを混合させることにより、アクセスの効率化をはかる。

ファイル構成の詳細は別の機会にのべる。

DMLの処理は、ファイル情報代数系の上に翻訳して、ファイル情報代数系を用いて行なう。

第3レベルにおいては、ユーザはスキーマを知らず、外部スキーマと定義することもせず、質問中に適当にスキーマを記述しながら行う。その度DDLのaliasを参照しながら、ユーザの意図する個体が配置されてゆく空間をさがしてゆく。構造がユーザの想定したものと全く同じでなくても、位相情報空間を任意の超平面に射影することによりユーザのスキーマと合うならば、それに合わせて処理を行う。

PRDBMSに関しては Conceptual Schema-free DBMS を用いる。

## [VI] むすび

位相情報空間データベースマネージメントシステムの基礎となる位相情報空間モデルと，そのモデルに基づく位相情報空間型ＤＢＭＳの構成原理，およびその言語周辺についてのべた。ＤＤＬ，ＤＭＬともまだ完全ではないが，位相情報空間モデルの持つ表現能力（概念の近傍関係，包含関係等）を十分に生かした言語にする為に，概念間の様々の関係を検索できるよう改善中である。更に他モデルで出来る処理も十分でき，かつSyntaxがあまり複雑にならないように検討しているが，微妙な所を表現しようとすればSyntaxが複雑になり，また記憶しなければならない予約語が増加する。逆に簡単化すれば能力が低下し，そのtrade-offが大きな問題となっている。

発表関連論文

位相情報空間モデル　：電子通信学会論文誌　Ｊ６３-Ｄ１
１９８０年１月号　pp ７９－８６
Schema-free DBS & ファイル情報代数系　：電子通信学会論文誌　Ｊ６４-Ｄ３，'８１年３月号　pp １９０－１９７
画像データベースシステム：　情報処理学会論文誌
２３．２．　'８２年３月号　pp １１６－１２３